论 文　SKA 区域中心数据处理系统和科学应用

低频射电脉冲星搜索的性能优化方法

韦建文[1]，张晨飞[1]，张仲莉[2,3,4*]，余婷[2,5]，林新华[1]，安涛[2,3]

1. 上海交通大学网络信息中心, 上海 200240;
2. 中国科学院上海天文台 SKA 区域中心联合实验室, 射电天文重点实验室, 上海 200030;
3. 鹏城实验室 SKA 区域中心联合实验室, 深圳,518066;
4. 上海引力波探测前沿科学中心, 上海 200240;
5. 中国科学院大学天文与空间科学学院, 北京 100049
* 联系人, E-mail: zzl@shao.ac.cn

收稿日期: 2022-06-28; 接受日期: 2022-07-xx;
中国 SKA 专项（编号: 2020SKA0120200）、国家重点研发计划 (编号: 2018YFA0404603)、国家自然科学基金资助项目（批准号: 12041301,11873079）和中国科学院青年创新促进会项目 (编号:2021258) 资助项目

摘要　随着平方公里阵列射电望远镜（SKA）等大科学装置的建设和运行, 以及大数据和高性能计算创新平台的提出, 天文学与高性能计算之间的联系日趋紧密. 天文学计算, 特别是作为 SKA 的主要科学方向之一的脉冲星搜索, 具有数据量大、计算量多的特点. 本文介绍了一种基于 OpenMP 多线程和多进程技术来加速脉冲星搜索管线的方案, 提出了一种解决负载不平衡问题的方法, 并成功的将优化管线安装于中国 SKA 区域中心原型机的 x86 和 ARM 计算节点. 通过默奇森大视场阵列望远镜（MWA）的脉冲星观测数据搜寻测试, 与原始单线程方法相比, 管线分别获得 10.4–12.2 和 24.5–27.6 倍的加速比. 其中 ARM 平台比 x86 平台的计算快 1.1–1.3 倍, 显示出其在 SKA 数据处理方面的巨大潜力. 在中国 SKA 区域中心原型机上部署的脉冲星优化搜索管线, 近期将重点应用于 MWA 南天快速两米巡天（SMART）项目的低频脉冲星搜寻, 以满足包括引力波探测计时阵在内的多种科学需要.

关键词　平方公里阵列, 脉冲星, 脉冲星搜索, 高性能计算, 并行优化

PACS: 47.27.-i,47.27.Eq,47.27.Nz,47.40.Ki,47.85.Gj

1　引言

脉冲星是具有强磁场的快速旋转中子星, 其典型半径约为 10-12 千米, 质量约为 1.4 倍太阳质量 $M_\odot$ [1]. 它于 1967 年在低频射电观测中首次被发现 [2], 在时域天文学、致密天体结构、星际介质模型, 以及引力波探测等领域有着重要价值. 观测研究显示, 脉冲星的辐射源于磁极, 当自转轴与磁轴存在夹角时, 就会产生灯塔效应, 使得地球上的观测者能够稳定地接收到周期性的脉冲辐射信号.

脉冲星在天文学和物理学上的巨大应用价值是建立在大样本基础上的, 而脉冲星的搜寻工作已经在全球范围内展开. 例如美国 Arecibo 望远镜的银盘 L 波段馈电阵列脉冲星巡天（PALFA） [3] 和澳大利亚 Parkes 望远镜的高时间分辨率宇宙脉冲星巡天（HTRU） [4,5]. 我国自主研制的 500 米口径球面射电望远镜“天眼”FAST [6–8], 至今已发现 350 多颗新脉冲星（如[9–12]），以及 FAST 多科学目

标同时扫描巡天（CRAFTS）中的 200 多颗脉冲星候选体[13,14] 1). 同时, 几百 MHz 以下的低频脉冲星探量也在进行中. 例如美国的 GBNCC 巡天项目[15] 利用 Green Bank 射电望远镜在 350MHz 对南纬 40° 以北天区进行全面搜寻, 已发现了 195 颗脉冲星2). Arecibo 在 327MHz 的巡天, 自 2010 年起, 已发现了 96 颗脉冲星[16,17]. 荷兰的 LOFAR 低频阵列射电望远镜[18] 在更低的 135MHz 频率上发现了 73 颗脉冲星[19,20]. 迄今为止, 已发现超过 3300 颗射电脉冲星（ATNF 目录[21]）. 它们大部分分布在银盘附近, 只有一小部分分布在高银纬地区, 这与脉冲星是由超新星爆炸产生的理论是一致的.

正在建设的平方公里阵列（Square Kilometre Array, SKA）射电望远镜的第一阶段（SKA1, 整个规模的 10%）预期能够发现 18000 多颗脉冲星, 超过目前已知数目的 5 倍[22]. 搜寻更多的脉冲星需要更长的积分时间, 而计算能力的需求是随着观测时间的立方增长的, 因此, 提高脉冲星搜寻速度是一个亟待解决的问题. 如今, 有多项工作利用现代多核处理器来优化脉冲星的搜索. Yu 等人[23] 测试了开启和关闭超线程、不同线程数对脉冲星探测和搜索工具 PRESTO[24]（Scott Ransom 开发）程序性能的影响, 但其代码还可以进一步优化. Dimoudi 等人[25] 在 GPU 上使用傅立叶空间加速搜索 (FDAS) 算法来优化脉冲星搜索, 但是这个 GPU 算法尚未集成到 PRESTO 中. Wang 等人[26] 使用 FPGA 芯片来加速脉冲星搜索, 而 FPGA 芯片没有被天文学家广泛应用于科学数据处理, 也不能与 PRESTO 很好的嵌合. 相比之下,x86 或 ARM 架构的多核 CPU, 支持 95% 以上的工作负载, 是我们加速脉冲星搜索的理想平台.

由于脉冲星的频谱陡峭, 理论上更容易在较低频率探测到脉冲星, 因此低频射电观测对于寻找新的脉冲星至关重要 [22]. 此外, 低频望远镜具有更大的探测视场（FoV）. 但由于低频环境噪声的干扰和星际介质的散射效应, 使得低射频脉冲星的探测变得困难. 随着 SKA-low（SKA 的低频阵列[27]）先导项目的运行, 脉冲星的低频搜索技术在不断进步. SKA-low 将成为最先进的脉冲星低频探测望远镜之一.

在 SKA 建成之前, 许多 SKA 的探路者设备已经投入使用. 位于西澳大利亚的默奇森大视场阵列望远镜（MWA）[28] 是 SKA-Low 的先导项目, 其视场包括具有最密集脉冲星分布的银河系中心区域. MWA 的电压捕获系统（VCS）[29] 覆盖 MWA 70–300 MHz 的频率范围, 在 30.72 MHz 的总带宽上细分了 3072 个频率通道, 因此具备 0.1 kHz 的频率分辨率. 在 0.1 ms 的时间采样下, 每小时的观测将产生 28 TB 的数据. 虽然数据量巨大给后随的计算处理带来不小的挑战,MWA 依然预期探测到 230 颗新的脉冲星[30]. 而未来的 SKA1-Low 则预期可发现大约 11,000 颗脉冲星[22].

在 SKA 时代, 单碟射电望远镜灵敏度和巡天效率难以提高的缺点将得到彻底克服. SKA 有望获得前所未有的高灵敏度、大视场和高巡天速度, 同时产生海量观测数据[27,31]. 当 SKA1 完成后, 数据生成速率最高可达每秒 TB 的级别[32,33]. SKA1 生成的海量数据将需要每秒至少 300Pflops 浮点运算的运算能力[31], 这几乎相当于当前世界上最快的超级计算机的能力(例如 Fugaku 的最大性能为 442 PFlops[34]）. 考虑到传输和计算效率, 实际对数据处理的需求将大大超过这个理论估计.

顺利完成海量数据的处理和存储是 SKA 正常运行的前提, 因此 SKA 主要成员国正计划建立各自的区域中心（SRC）, 来执行科学数据深度分析和数据产品长期存储的任务, 并且支撑全球科学用户的工作[35]. 分布式 SRC 将形成一个全球的协作和协调网络, 负责 SKA 的科学数据产品[36]. 全球 SRC 网络的设计、开发、运营和维护符合 SKA 天文台需求和国家/地区的战略发展.

上海天文台正牵头推进中国 SKA 区域中心原型系统建设（CSRC-P）, 并且已建成世界第一台

1) FAST 脉冲星目录可以在以下链接中获得: (*a*)https://crafts.bao.ac.cn/pulsar/; (*b*)https://fast.bao.ac.cn/cms/article/65/; (*c*)https://zmtt.bao.ac.cn/GPPS/

2) http://astro.phys.wvu.edu/GBNCC/

3) https://www.skatelescope.org/news/china-ratifies-skao-convention/

SRC 原型机[37,41]3)及其软件系统[39]. SKA 区域数据中心建成后, 将以 MWA 为基础, 同时开拓和 ASKAP、MeerKA0T 和 LOFAR 的合作, 开展 SKA 先导项目的科学预研究和技术发展, 其中包括低频射电连续谱巡天观测项目[40], 以及本文所开展的低频射电脉冲星搜索研究. 在全面建设 SRC 之前, 原型机的研制和测试至关重要. CSRC-P 承担了 SKA 先导和探路望远镜的数据处理技术验证、数据挑战测试、数据处理流程开发和数据分析的任务. CSRC-P 采用混合异构计算架构的设计, 同时使用 x86 和 ARM[42] 处理器集群, 它是第一个使用 ARM 架构的 SKA 数据处理系统. 正如 2019 年的 SKA 工程会议4)上所展示的,ARM 计算节点的基准测试显示了出色的整体性能. CSRC-P 配备了多核处理器和出色的存储和网络系统, 为改进脉冲星搜索管线提供了理想的测试平台.

本文将介绍在现代多核处理器平台上针对脉冲星搜索管线执行的优化方法, 尤其是针对 CSRC-P 的设计, 以及针对 MWA-VCS 的四个非相干叠加数据进行搜寻测试的结果. 文章结构如下: 第2章为我们所采用的脉冲星搜寻管线, 第3章介绍我们对该管线中热点步骤的优化方法, 第4和第5章分别为优化管线的性能评估和总结. 值得一提的是, 该管线近期将重点应用于 MWA 的南天快速两米巡天 (SMART[43]) 项目的低频脉冲星搜寻当中.

2 脉冲星搜索流程

Scott Ransom 开发的 PRESTO[24] 是使用最广泛的脉冲星搜索管线.PRESTO 的最初设计目的是从对球状星团的长时间观测中有效地搜索毫秒脉冲双星. 目前, 已利用 PRESTO 搜索到了 600 多颗脉冲星, 其中包括 230 多颗毫秒脉冲星和毫秒脉冲双星[44]. PRESTO 代码主要由 ANSI C 语言编写, 还包含许多使用 Python 语言编写的例程. 使用 PRESTO 搜索脉冲星的管线如图1所示, 共有 6 步: 1) 消除射频干扰 (Remove Narrowband Radio Frequency Interference); 2) 消色散 (De-dispersion); 3) 傅里叶变换及消除红噪声 (FFT & Remove Red-noise); 4) 加速搜索 (Accelerate Search); 5) 候选体折叠 (Candidate Folding); 6) 单脉冲搜索 (Single Pulse Search). 而最终对于周期性和单脉冲的候选体筛选, 暂时采用人工的方法. 后续将根据不同的观测数据和需要在该步骤安插人工智能筛选程序. 在接下来的章节中, 我们将介绍筛选前 6 个步骤的方法, 以及为这些步骤设计的并行优化方案.

4) https://indico.skatelescope.org/event/551/

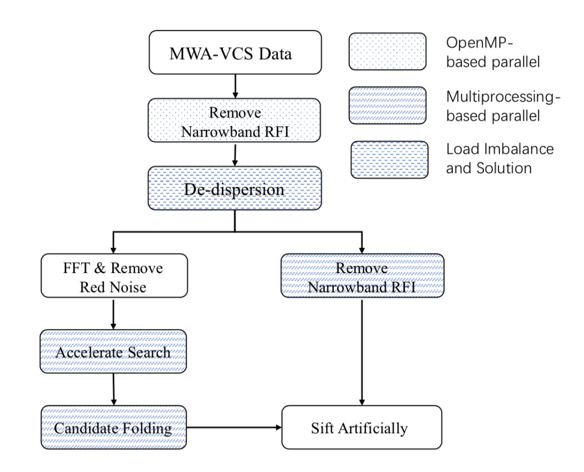


图 1　脉冲星搜索管线和分步优化方法

Figure 1　The flowchart of the pulsar search pipeline and the optimization of each step

2.1 消除射频干扰

射频干扰主要来源于人造射电源, 很容易被错误识别为脉冲星候选体, 而真正的脉冲星信号又会因为受到干扰而被忽略. 射频干扰对脉冲星的搜索与观测会产生严重的影响, 从而大幅增加后续流程的工作量.

PRESTO 提供了 rfifind.py 的程序, 可用于识别和去除很强的窄带射频干扰, 以及持续时间较短的宽带的射频干扰. 其算法将时域分为几秒的区间, 以满足大部分观测数据去除干扰的需要 (本文中测试设置为 12s). rfifind 用这个区间去扫描整个数据, 找出其中的射频干扰并标记, 生成标记.mask 文件.

另一个需要搜索的是周期性干扰信号. 由于射电信号必然会受到星际介质的影响, 所以色散量 (DM) 为 0 的信号是不存在的, 因此这类信号可以

视为射频干扰. PRESTO 对数据进行一次 DM-0 搜索, 找出色散量为 0 的信号, 并将搜索信息记录到.bird 文件中, 从而消除此类干扰对后续搜索的影响.

2.2 消色散

由于不同频率的电磁波在星际介质中的传播速度不同, 射电望远镜接收到的不同频率的信号会有时间差, 从而导致脉冲展宽. 频带越宽的射电望远镜在观测脉冲星时灵敏度越高, 但星际介质引起的色散影响也越大. 我们需要对观测到的数据进行消色散处理.

我们利用基于 PRESTO 的 prepsubband 程序进行消色散处理. prepsubband 程序执行消色散的过程如下: 根据特定的色散量 DM, 将不同通道组的信号在时间上进行移动, 最终生成一系列不同色散量的时间序列. 消色散操作会去除上一步标记的射频干扰. 对于信号较强的脉冲星, 消色散之后便会显现出脉冲信号. 若数据中存在色散量在搜索范围内的脉冲星信号, 其周期可以在后续搜索过程中被确定.

2.3 傅里叶变换及消除红噪声

得到不同色散量的时域数据后, 我们对序列进行离散傅里叶变换, 再从频域中展开搜索. 本文使用在计算机上能够高效实现离散傅里叶变换的算法: 快速傅里叶变换（FFT）, 从而极大提高脉冲星搜寻的效率.

由于环境中无处不在的粒子布朗运动, 采集到的信息会混有红噪声, 导致频域信号低频端升高. 因此, 对于傅里叶变换后的.fft 文件, 我们需要执行 PRESTO 中的红噪声消除程序以白化信号, 生成新的去除红噪声的.fft 文件.

2.4 加速搜索

在频域中, 一颗真实脉冲星的信号并非只在一个频率处出现, 而是表现为一个基频和一系列谐波. 为了充分利用谐波, 我们使用“谐波叠加技术”来加强脉冲星的频域信号, 并获取脉冲星的候选体.

PRESTO 软件的 accelsearch 程序提供了脉冲星的加速搜索算法, 可同时用于对普通脉冲星和双星系统中的脉冲星进行搜索. 其中,-numharm 参数可以设定最高谐波叠加次数. 理论上, 该参数设置得越高对搜索越有利, 但同时运算量也会成倍增加. 在本文中, 该参数设置为 16. accelsearch 中的-zmax 参数, 指的是在搜索中使用的参数空间中, 由双星系统的轨道运动引起的频域上的最大漂移量. zmax 的设定值越大, 运算耗时越长. 在本文中,zmax 的值设为 200.

2.5 候选体折叠

对于程序筛选出来的候选体, 我们在搜索到的色散量 DM 和周期 P 上将它们的时间序列进行折叠, 并分析其折叠后的轮廓.

基于 PRESTO 软件的 prepfold 有两种方法来折叠脉冲轮廓: 一种是在输入的 DM 和 P 值的附近进行小范围搜索后折叠, 目的是找到更精确的 DM 和 P 值. 另一种方式是直接采用输入的 DM 和 P 值进行折叠. 由于本文中对流程的优化目的是为大量的巡天搜寻做准备, 我们对每个候选体采用的是直接折叠的方式.

2.6 单脉冲搜索

部分脉冲星的信号不会表现出明显的周期性, 例如存在零脉冲、巨脉冲等现象的脉冲星. 具有零脉冲的脉冲星会在一段时间内没有脉冲信号; 有巨脉冲的脉冲星, 其巨脉冲可以达到平均脉冲强度的 100–1000 倍. 这两类脉冲星都很难在频域上被搜索到. 单脉冲搜索通常采用时域上的搜索算法. PRESTO 软件提供了单脉冲搜索程序, 可以得到不同色散量时间序列中较明显的单脉冲信号.

3 脉冲星搜索并行优化

3.1 脉冲星搜索管线热点分析

脉冲星搜索各步骤的时间占比如图 2所示. 其中, 消除射频干扰（Remove Narrowband RFI）、消色散（De-dispersion）、加速搜索（Accelsearch）、候选体折叠（Candidate Folding）以及单脉冲搜索（Single Pulse Search）是程序热点, 运行耗时高, 优化潜力大. 快速傅里叶变换（FFT）和消除红噪声（Remove Rednoise）耗时少, 优化潜力小, 不需要进行优化.

根据 PRESTO 的代码热点和程序编写语言分析, OpenMP 多线程和 Python 多进程方法是合适的优化方法, 具体理由如下：第一, 大部分代码是单核运行, 通过利用现代多核计算节点, 预期可以获得数倍的加速效果. 第二, 脉冲星多频道数据间没有依赖, 因此可以更好地并行化, 进一步缩短运行时间. 第三, 相对于多节点并行优化, 多线程或多进程的并行优化更易完成, 且正确性更容易验证, 因此可以作为未来优化的良好起点.

本文使用的并行技术如下所示, 分别为 Multiprocessing、OpenMP 以及负载平衡方法. 这些优化方法应用到的步骤如图 1所示.

- Multiprocessing：一种支持多种计算机语言的进程级并行技术. 本文使用 Python 的 multiprocessing 库 [45] 实现. 该库使用子进程而非线程, 从而能充分利用集群上的多核性能.

- OpenMP [46]：多核处理器并行编程的一种共享内存标准. 通过共享内存, 不同的核可以从共享内存中获取该核需要的数据用于处理.

- 负载平衡 [47] ：由于所有的核不可能同时完成工作, 部分核停止运行时, 仍可能有另一部分核在运行, 而空闲的核心会造成性能浪费. 因此我们需要均衡所有核心的负载, 尽可能避免有核心闲置.

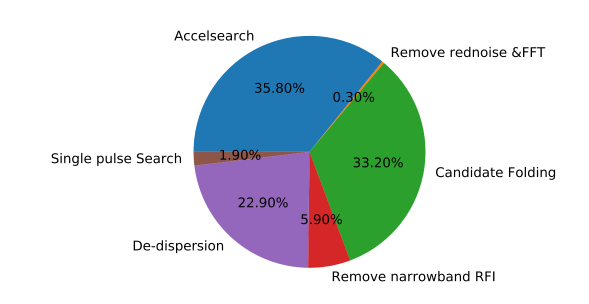


图 2 第4章样例 1 中脉冲星搜索管线热点分析
Figure 2 The hotspots of Pulsar Search Pipeline with case 1 in section 4

3.2 使用 OpenMP 优化窄带射频干扰

我们使用 OpenMP 来优化窄带射频干扰消除的过程. 窄带射频干扰消除可分为两步：搜索射频干扰和写.mask 文件. 原程序会调用 rfifind, 将时间序列分割为多个固定秒数的区间, 随后在每个区间中搜索射频干扰（算法 1）. 因为这些区间中没有数据依赖, 我们将 OpenMP 的 pragma 制导语句添加在 rfifind 的 C 语言源码中, 来并行化搜索射频干扰（算法 2）. 而将搜索结果写入 mask 文件的过程耗时较少, 无需进一步优化.

算法 1 原始的窄带射频干扰消除

Require: M-频道数量
Ensure: 消除窄带射频干扰后的频道
1: for $i = 0$ to $M - 1$ do
2: 　调用 Search_rfifind 来搜索第 i 个 channel 的射频干扰, 并将结果写入 $data[i]$ 中
3: 　将 $data$ 写入文件中
4: end for

表 1 低频 MWA 非相干叠加数据的消色散方案

Table 1 The de-dispersion scheme for MWA incoherent summed data at low frequency

Low-DM ($cm^{-3}pc$)	High-DM ($cm^{-3}pc$)	dDM ($cm^{-3}pc$)	Nsteps	Downsamp	nsub (ms)	Effective time resolution
1.0	22.9	0.02	1093	1	4	0.1
22.9	45.7	0.05	457	2	8	0.2
45.7	91.5	0.11	415	4	16	0.4
91.5	183.0	0.21	435	8	32	0.8
183.0	250.0	0.43	115	16	64	1.6

算法 2 优化后的窄带射频干扰消除

Require: M-频道数量,N-核心数量

Ensure: 消除窄带射频干扰后的频道

for $i = 0$ to $M/N - 1$ do
  for $j = 0$ to $N - 1$ do
    OpenMP 线程 j 调用 Search_rfifind 来搜索第 $iN + j$ 个 channel 的射频干扰, 并将结果写入 $data[j]$ 中
  end for
  for $j = 0$ to $N - 1$ do
    OpenMP 线程 j 将 $data[j]$ 写入文件中
  end for
end for

3.3 使用 Python 执行多进程并行

在去除射频干扰后每一步热点步骤都具有相似的计算模式：使用 Python 脚本输入多个独立的文件、输出同样写入到独立的文件中. 因此, 它们在下文中都可以使用并行优化. 本文使用 Python 的 multiprocessing 库, 把所有待处理的文件均分成 N 份, 每份数据交给一个 Python 进程绑定的 CPU 核心处理 (算法 3). 其中, 基于消色散方案, 加速搜索和单脉冲星搜索的并行流程有数千个, 而候选体折叠的并行流程通常也达到一百左右. 下面我们基于并行处理对每一步进行详细说明.

算法 3 多进程并行

Require: N ← 核心数,list ← 任务列表

for $i = 0$ to $N - 1$ do
  按照 list 将任务分配给第 i 个核心
end for
N 个核心同时执行其被分配的任务

3.4 消色散中的并行化与负载平衡

采用表 1 中所示的适用于 MWA 低频脉冲星搜寻的消色散方案, 我们将消色散的循环展开为两层循环, 并使用 OpenMP 进行优化. 消色散过程的热点部分为从观测到的数据中消除色散的循环：首先, 读取一条没有窄带射频干扰的数据并生成预处理过的数据（伪代码中的 GetData）. 然后, 使用前一个循环中的数据（上一条数据）和当前循环的数据（本条数据）来消除色散（伪代码中的 dedisp）. 其中, 由于第一次运行时不存在上一条数据, 因此在第一个循环中, 本条数据会被使用两次.

在消色散的循环中, 由于上一条数据需要用于本条数据的消色散, 因此存在数据依赖. 为了解决这一问题, 我们将循环展开为两个循环, 并分别做并行. 第一个循环预处理数据, 并将它们加载进数组中. 第二个循环读取预处理的数据, 并进行消色散的操作（算法 5）.

算法 4 原始的消色散流程

Require: M ← 频道数量,$data$ ← 数据集合

Ensure: 消色散后的数据

for $i = 0$ to $M - 1$ do
  $current \leftarrow data[i]$ 第 i 个频道的数据
  对 $current$ 做快速傅里叶变换
  if $i > 0$ then
    用 $prev$ 和 $current$ 做消色散
    对 $current$ 做快速傅里叶变换
  else
    使用两次 $current$ 做消色散
    对 $current$ 做快速傅里叶变换
  end if
  $prev \leftarrow current$
end for

算法 5 优化后的消色散流程

```
Require: M ← 频道数量,N ← 核心数量,data ← 数据集合
Ensure: 消色散后的数据
  for i = 0 to M/N − 1 do
    if i > 0 then
      prev ← current[N − 1]
    end if
    for j = 0 to N − 1 do
      current[j] ← data[iN + j] 第 iN + j 个频道的数据
      对 current[j] 做快速傅里叶变换
    end for
    for j = 0 to N − 1 do
      if i == 0 and j == 0 then
        用两次 current[0] 做消色散
        对 current[0] 做快速傅里叶变换
      else if j == 0 then
        用 prev 和 current[0] 做消色散
        对 current[0] 做快速傅里叶变换
      else
        用 current[j − 1] 和 current[j] 做消色散
        对 current[j] 做快速傅里叶变换
      end if
    end for
  end for
```

我们将消色散的循环展开为两层循环, 并使用 OpenMP 进行优化. 消色散过程的热点部分为从观测到的数据中消除色散的循环: 首先, 读取一条没有窄带射频干扰的数据并生成预处理过的数据（伪代码中的 GetData）. 然后, 使用前一个循环中的数据（上一条数据）和当前循环的数据（本条数据）来消除色散（伪代码中的 dedisp）. 其中, 由于第一次运行时不存在上一条数据, 因此在第一个循环中, 本条数据会被使用两次.

在消色散的循环中, 由于上一条数据需要用于本条数据的消色散, 因此存在数据依赖. 为了解决这一问题, 我们将循环展开为两个循环, 并分别做并行. 第一个循环预处理数据, 并将它们加载进数组中. 第二个循环读取预处理的数据, 并进行消色散的操作.

但是, 消色散中每条 prepsubband 命令分配一个 CPU 核心的策略可能导致负载不均衡. 如图3所示, 用于完成 prepsubband 的时间在 300 秒到 1500 秒不等. 提前完成任务的 CPU 核心必须等待最慢的核心完成后才能继续执行下一个任务. 表1列举了单个计算任务的差异导致的运行时间的不同. 表中 Low-DM 和 High-DM 分别代表消色散操作允许的下限和上限,dDM 代表 DM 步长,Nsteps 代表需要执行的步数,Downsamp 代表降采样系数,nsub 代表频率子通道的数量. 每条 prepsubband 命令执行 100 次 Nsteps 数, 并且消色散操作共有 28 条命令. 同时,nsub 的值越大, 命令花费的时间就越多.

为了平衡 prepsubband 的运行时间, 我们针对消色散操作设计了负载平衡算法. 使用 OpenMP 分配与每个任务的工作负载成比例的线程数量. 表2显示了在给定 nsub 和线程数时命令的预测执行时间.

我们根据命令耗时的预测值, 生成了局部最优的进程线程配置方案, 从而达到平衡进程负载的目的. 该配置算法包括: 总体的进程数, 每个进程包含的命令以及相应的线程数. 其中, 一个进程包含若干命令, 命令之间串行执行且使用相同的线程数. 具体步骤如下, 代码见算法6:

1. 根据消色散方案, 假设需要执行的命令数量为 C, 则初始化 C 个进程, 每个进程包含一条消色散命令, 且每个进程中所有命令的线程数为 1;

2. 根据命令耗时的预测值, 记 T 为所有进程耗时的最大值;

3. 如果当前使用的线程数小于等于节点上 CPU 核心数, 找出最耗时的进程, 使该进程的线程数加 1. 否则, 找出两个耗时最短的进程, 将这两个进程中的命令合并为一个进程, 并将其线程数设置为原来两个进程中线程数的较大值;

4. 更新 T 的值并与原值进行比较, 验证是否满足收敛条件; 若满足收敛条件且当前使用的线程数大于等于节点上 CPU 核心数, 则进程线程配置算法结束, 并输出消色散的配置方案, 否则, 返回前一步.

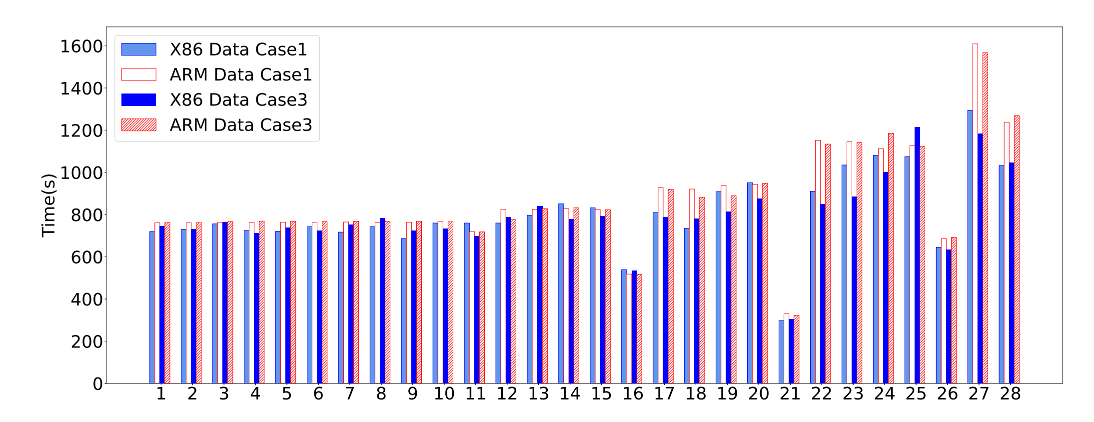


图 3 消色散操作在各核心上负载不均导致运行时间差异

Figure 3 Work imbalance leads to runtime variation on CPU cores of x86 and ARM

4 性能评估

我们选择非相干模式下, 中心频率为 185MHz 的 4 次 MWA-VCS 观测作为测试样例 (如表3所示). MWA-VCS 的非相干模式具有约 600 平方度的大可视天区, 非常适合 SMART 巡天前期, 大面积的浅层普查. 2019 年, 龚宏宇等人[48] 就发表了在 50 次 MWA-VCS 非相干数据巡天旁瓣中找到的北天脉冲星.

我们对这四次观测的搜寻实验都分别在 CSRC-P 的 x86 和 ARM 计算节点上运行 (具体配置见表4) . 这两种节点的硬件和软件配置基本接近, 适合用于本次实验结果的比较分析. 我们比较了 x86 和 ARM 平台上优化前和优化后的运行时间、结果的正确性及不同优化策略取得的优化效果.

算法 6 消色散中的负载平衡算法

Require: 所有命令的预测用时,
  $N \leftarrow$ 核数,$C \leftarrow$ 命令数
Ensure: 任务分配方案
  初始化 C 个进程, 每个进程包含一条消色散命令, 且线程数为 1
  $M \leftarrow C$ /*M 为当前线程总数 */
  $prevT \leftarrow 0$
  $currentT \leftarrow$ 当前耗时最长的进程需要的时间
  $\Delta T \leftarrow |prevT - currentT|$
  $iter \leftarrow 0$
  while $\frac{\Delta T}{currentT} \geqslant 5\%$ and $iter < N$ or $M > N$ do
    if $M \leqslant N$ then
      找出运行用时最大的进程 a
      将该进程的线程数加 1
      $M \leftarrow M + 1$
    else
      找出运行用时最小的两个进程 a,b, 线程数分别为 t_a,t_b
      合并 a,b 为一个进程, 取 $\max\{t_a, t_b\}$ 为新进程的线程数
      $M \leftarrow M - \min\{t_a, t_b\}$
    end if
    $prevT \leftarrow currentT$
    $currentT \leftarrow$ 当前耗时最长的进程需要的时间
    $\Delta T \leftarrow |prevT - currentT|$
    $iter \leftarrow iter + 1$
  end while

表 2 每条 prepsubband 命令在给定 nsub,Nsteps 及 OpenMP 线程数下的预估运行时间

Table 2 The assumed runtime of each prepsubband command given nsub,Nsteps and the number of OpenMP threads

	1 thread	N threads
nsub = 4ms	$700 \times \frac{Nstep}{100}$	$1.25 \times \frac{700\times\frac{Nstep}{100}}{N}$
nsub = 8ms	$800 \times \frac{Nstep}{100}$	$1.25 \times \frac{800\times\frac{Nstep}{100}}{N}$
nsub = 16ms	$900 \times \frac{Nstep}{100}$	$1.25 \times \frac{900\times\frac{Nstep}{100}}{N}$
nsub = 32ms	$1000 \times \frac{Nstep}{100}$	$1.25 \times \frac{1000\times\frac{Nstep}{100}}{N}$
nsub = 64ms	$1200 \times \frac{Nstep}{100}$	$1.25 \times \frac{1200\times\frac{Nstep}{100}}{N}$

4.1 正确性验证

该管线的设计目的是从时间序列数据中寻找脉冲星候选体. 搜寻流程完成后, 关于脉冲星候选体的信息存储在 candslist.txt 中, 其中包含每颗脉冲星候选体的色散和周期. 我们发现不管在哪种节点上运算, 经过优化的搜寻管线和原管线获得了完全一致的结果, 以此验证了我们优化的脉冲星搜寻管线的正确性.

4.2 OpenMP 并行性能分析

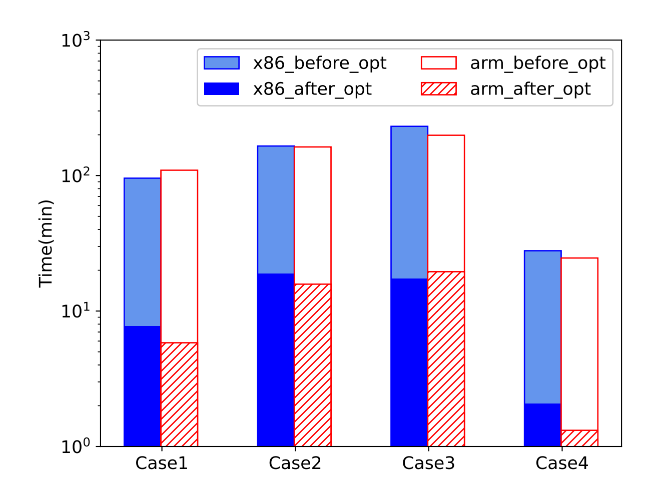


图 4 x86 及 ARM 单节点上消除射频干扰的性能比较

Figure 4 The performance comparison of *Remove Narrow–band RFI* on x86 and ARM's single node

表 5 消除射频干扰的加速比和并行效率

Table 5 The speedup and parallel efficiency of *Remove Narrow–band RFI*

Case	1	2	3	4
x86 加速比	12.4	8.7	13.3	13.4
x86 并行效率	44.2%	31.1%	47.6%	47.8%
x86 优化后运行时间	7.7s	18.9s	17.3s	2.1s
ARM 加速比	18.8	10.3	10.2	18.7
ARM 并行效率	19.6%	10.7%	10.6%	19.5%
ARM 优化后运行时间	5.8s	15.8s	19.5s	1.3s

我们在 x86 和 ARM 单节点上测试了消除射频干扰的 OpenMP 并行性能. 测试结果见图4. 表5显示了在不同节点上不同数据上, 消除射频干扰的加速比 S_p 及并行效率 ϵ_p, 其计算公式分别为 $S_p = T_1/T_p$ 和 $\epsilon_p = T_1/pT_p$. 其中,p 为计算节点的核数,T_1 为单核运行程序的耗时,T_p 为使用 p 核运行程序的耗时.

对于不同的测试数据,ARM 和 x86 具有相似的加速比. 单个 ARM 通常比 x86 节点拥有更多的 CPU 核心, 因此消除射频干扰在 ARM 集群上运行速度更快. 但是 ARM 和 x86 集群上的加速比不完全等同于计算核数目的比例. 这是因为 OpenMP 的多线程性能受到内存带宽的限制, 这导致 ARM 集群的并行效率不会随着 CPU 核数的增加而线性增加.

表 3 非相干模式下的 MWA-VCS 观测脉冲星搜寻测试样例
Table 3 The MWA-VCS observations in incoherent mode for pulsar search tests

Case	ObsID	R.A.(J2000)	Dec.(J2000)	Duration(s)	Pulsar searched
1	1088850560	13:20:07.2000	-26:37:12.0000	3535	-
2	1145367872	11:34:17.2956	-33:25:06.9706	3613	J1116-4122
3	1115381072	10:10:08.1228	+10:39:45.9377	4868	J0953+0755
4	1131415232	14:39:27.4317	-71:09:36.8236	594	J1430-6623,J1453-6413,J1456-6843

表 4 软硬件配置
Table 4 Software and hardware configuration

节点	CPU	内存	操作系统	内核版本	软件环境
x86	Intel Xeon Gold 45367872/case2/gc (28cores 2.60GHz)	6-Channel DDR4-2666	CentOS 7	3.10.0	Python:2.7 gcc:8.3
ARM	Kunpeng 920 CPU (48cores 2.50GHz)	8-Channel DDR4-3200	CentOS 7	4.14.0	Python:2.7 gcc:8.3

4.3 多进程并行性能分析

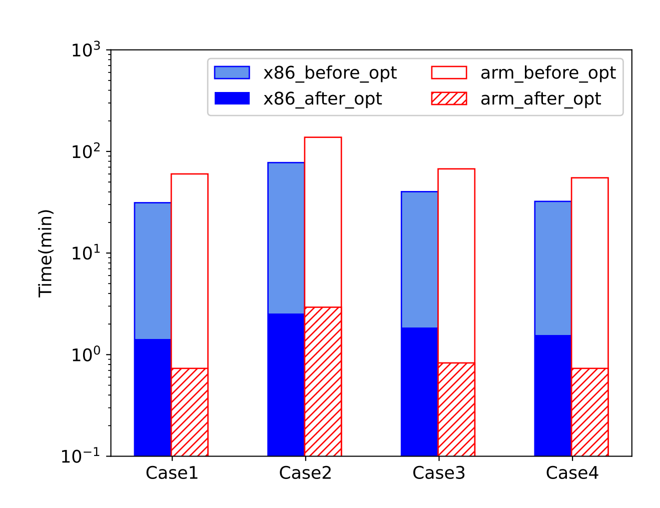


图 5 x86 和 ARM 单节点上的单脉冲搜索性能比较
Figure 5 The performance comparison of *Single Pulse Search* on x86 and ARM's single node

表 6 单脉冲搜索的加速比和并行效率
Table 6 The speedup and parallel efficiency of *Single Pulse Search*

Case	1	2	3	4
x86 加速比	22.1	30.7	21.8	20.8
x86 并行效率	78.8%	109.5%	77.7%	74.3%
x86 优化后运行时间	1.4s	2.5s	1.9s	1.6s
ARM 加速比	82.0	47.0	81.2	75.0
ARM 并行效率	85.5%	48.9%	84.6%	78.1%
ARM 优化后运行时间	0.7s	2.9s	0.8s	0.7s

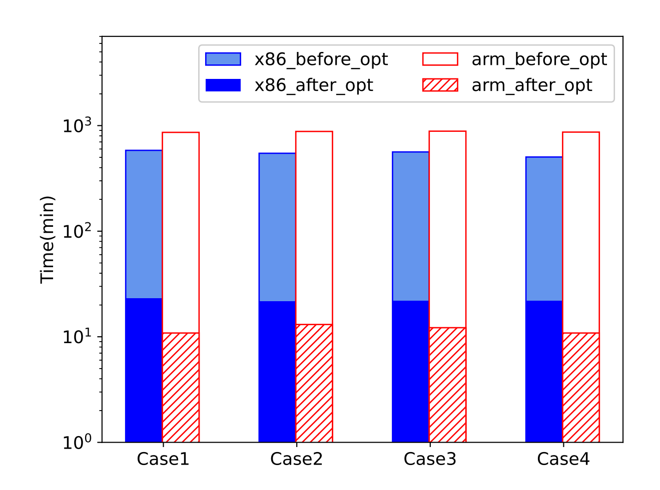


图 6 x86 和 ARM 单节点上的加速搜索性能比较
Figure 6 The performance comparsion of *Accelerate Search* on x86 and ARM's single node

表 7 加速搜索的加速比及并行效率
Table 7 The speedup and parallel efficiency of *Accelerate Search*

Case	1	2	3	4
x86 加速比	25.2	25.4	25.6	23.0
x86 并行效率	89.9%	90.6%	91.5%	82.0%
x86 优化后运行时间	23.2s	21.6s	22s	21.9s
ARM 加速比	79.6	67.4	72.7	79.8
ARM 并行效率	82.9%	70.2%	75.7%	83.1%
ARM 优化后运行时间	10.9s	13.1s	12.2s	10.9s

基于前文所述的多进程并行方法, 我们在 x86 和 ARM 单节点上测试了使用 multiprocessing 并行化单脉冲搜索、加速搜索、候选体折叠和消色散的并行性能.

单脉冲搜索使用了基于 PRESTO 的 Python

脚本 single_pulse_search.py. 性能比较见图5, 加速比及并行效率见表6. 加速搜索使用了 PRESTO 中的 accelsearch. 其中, 谐波叠加上限 numharm 设置为 16, 频域最大偏移 zmax 设置为 200. 表7及图6显示了在不同节点上运行不同数据的加速比和并行效率. 从图中可以发现, 单脉冲搜索和加速搜索都能取得很高的并行效率. 这是因为它们对数千个文件做处理, 且文件之间的处理没有依赖, 因此多进程的方法可以很好地提升它们的性能.

对于候选体折叠, 我们运行了基于 PRESTO 的 prepfold, 折叠后作图点数 n 设置为 128. 表8及图7显示了不同节点上运行不同数据的加速比和并行效率. 候选体折叠会执行多条不同参数的 prepfold 命令, 这些命令的耗时相同. 观测数据 case1 需要执行 50 条 prepfold 命令, 在 28 核的 x86 并行运行时, 先运行 28 条命令, 完成后再运行剩下的 22 条. 此时会存在空闲的 CPU 核心, 从而降低并行效率. 在 96 核心的 ARM 单节点上运行时, 会有 46 个空闲的 CPU 核心, 因此并行效率会较低. 观测数据 1091309464 需要执行 88 条 prepfold 命令, 此时 ARM 上由于命令数量变多, 并行效率会得到提升. 而在 x86 上, 由于命令数量的上升, 运行过程中整个节点的 CPU 核心被占满的时间比例会增加, 所以并行效率也会得到提升.

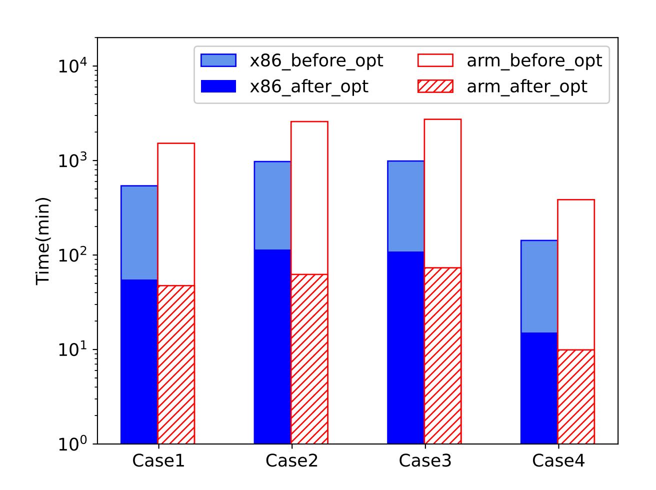


图 7 x86 及 ARM 单节点上的候选体折叠性能比较
Figure 7 The performance comparison of *Candidata Folding* on x86 and ARM's single node

表 8 候选体折叠的加速比及并行效率
Table 8 The speedup and parallel efficiency of *Candidata Folding*

Case	1	2	3	4
x86 加速比	9.8	8.6	9.1	9.5
x86 并行效率	34.9%	30.7%	32.7%	33.9%
x86 优化后运行时间	55.2s	113.6s	108.7s	15.1s
ARM 加速比	32.1	41.5	37.3	38.8
ARM 并行效率	33.4%	43.2%	38.8%	40.5%
ARM 优化后运行时间	47.6s	62.4s	73.4s	9.9s

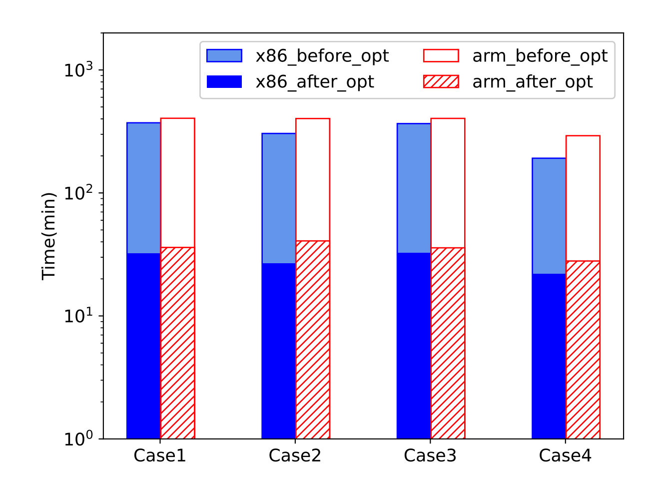


图 8 x86 和 ARM 单节点上的消色散性能比较
Figure 8 The performance comparsion of *De − dispersion* on x86 and ARM's single node

表 9 消色散的加速比及并行效率
Table 9 The speedup and parallel efficiency of *De − dispersion*

Case	1	2	3	4
x86 加速比	11.5	11.4	11.3	8.7
x86 并行效率	41.0%	40.6 %	40.3%	31.0%
x86 优化后运行时间	32.4s	26.8s	32.5s	22.0s
ARM 加速比	11.2	9.9	11.3	10.4
ARM 并行效率	11.7%	10.3%	11.8%	10.9%
ARM 优化后运行时间	36.1s	40.8s	35.8s	292.2s

最后, 对于消色散, 我们运行了基于 PRESTO 的 prepsubband, 并根据 MWA 对非连续低频数据的消色散方案设置运行参数. 不同节点上不同数据的加速比及并行效率见表9及图8. 消色散的并行效率同样较低, 因为根据消色散方案给出的参数空间, 在多核计算节点并行 prepsubband 命令的时候, 命令之间的耗时差异较大, 从而造成运行时的计算资

源浪费以及并行效率的降低. 在下一节中, 我们针对 ARM 节点采用 OpenMP+multiprocessing 组合的方式, 来尽可能消除其负载不平衡的问题, 以充分利用每节点的 96 个核心.

4.4 负载平衡优化

根据图 2 中消色散命令的耗时分布, 我们可以发现, 命令 1-11 对应 $nsub$ = 4 的情况,12-16 对应 $nsub$ = 8 的情况,17-21 对应 $nsub$ = 16 的情况,22-26 对应 $nsub$ = 32 的情况,27-28 对应 $nsub$ = 64 的情况, 其中 16、21 和 26 的 Nsteps 分别为 57、15 和 35, 因此它们的耗时会较少.

针对 x86 单节点 28 个 CPU 核心的特性, 我们采用负载平衡生成算法所获得的优化并行策略, 是使用 multiprocessing 运行 14 个进程, 每个进程中串行运行 1-3 条 prepsubbands 命令, 每条命令使用 2 个 OpenMP 线程来运行, 这样就可以用满 x86 单节点上的 28 个 CPU 核心, 最大化地平衡每个进程的负载. 命令与进程的对应关系如下: $(1, 2, 21), (3, 16, 26), (4, 25), (5, 24), (6, 23), (7, 22), (8, 20), (9, 19), (10, 18), (11, 17), (12, 15), (13, 14)$. 其中, 每个括号代表一个进程, 括号中的数字代表该进程中串行运行的命令编号.

表 10 ARM 上消色散的负载平衡策略

Table 10 The load balance strategy of De-dispersion on ARM

命令 id	OpenMP threads 数量
1-16	2
17-20	4
21	2
22-25	6
26	2
27-28	10

受限于流水线的算法设计, 当前启用的 28 个进程无法充分发挥 ARM 节点 96 个核心的计算能力. 我们设计了 Multiprocessing+OpenMP 多线程的两级并行方法, 为不同的进程分配 2-10 个 OpenMP 线程, 从而充分发挥 96 核心计算能力. 每个进程根据运行命令中的 nsub 的值分配不同的 OpenMP 线程数, 对应关系如表10所示.

通过上述的负载平衡策略, 我们比较了进一步优化前后的性能对比, 结果见图9和表11. 我们可以发现,ARM 的提速效果更明显, 因为在负载平衡优化前,ARM 单节点只使用了 28 个进程来运行消色散, 而负载平衡后, 每个进程都会使用多个 OpenMP 线程, 从而用满了 96 个 CPU 核心. 而 x86 在负载平衡前后都已经用满了 28 个 CPU 核心, 所以提升效果不及 ARM.

四个测试数据的整体加速比见图10及表12. 相较于单线程方案, 我们在 x86 平台上实现了 10.4–12.2 倍的加速, 在 ARM 平台上实现了最高 27.6 倍的加速. 在本文的测试数据中,ARM 平台的运行时间相比 x86 平台短 1.1–1.3 倍. 这是由于 ARM 节点有更多的运算核心.

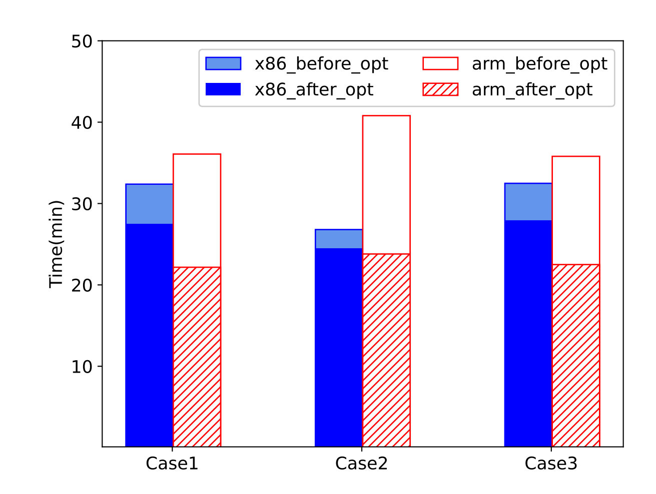


图 9 x86 及 ARM 节点消色散运行时间比较

Figure 9 The runtime comparsion of de-dispersion on the x86 and ARM nodes.

表 11 使用负载平衡策略后的消色散并行效率比较

Table 11 The parallel efficiency comparsion of De-dispersion after using load balancing method

Case	1	2	3
x86 负载平衡前	41.0%	40.6%	40.3%
x86 负载平衡后	48.3%	44.4%	46.9%
ARM 负载平衡前	11.7%	10.3%	11.8%
ARM 负载平衡后	19.0%	17.6%	18.7%

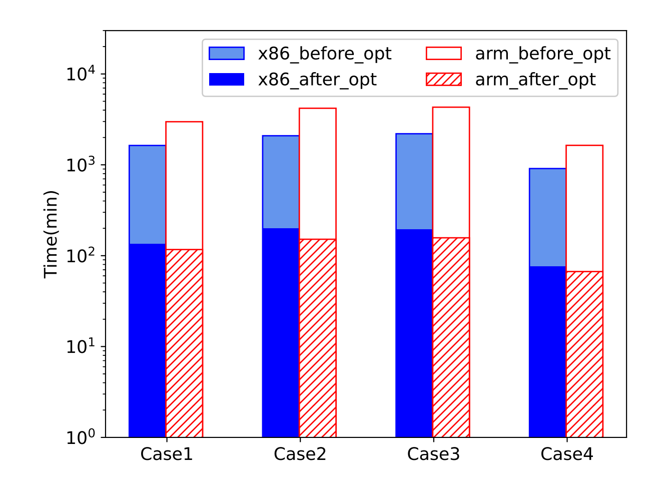


图 10 x86 和 ARM 单节点上整体管线的性能比较
Figure 10 The performance comparison of the whole pipeline on x86 and ARM's single node.

表 12 x86 及 ARM 的整体管线加速比
Table 12 The speedup of the whole pipeline on x86 and ARM

Case	1	2	3	4
x86 优化后运行总时间	133.6s	199.9s	195.4s	76.3s
x86 加速比	12.2	10.4	11.3	11.9
ARM 优化后运行总时间	117.0s	151.7s	157.3s	67.0s
ARM 加速比	25.8	27.6	27.4	24.5
ARM 相较于 x86 的加速比	1.1	1.3	1.2	1.1

5 总结

本文基于 OpenMP 多线程和 multiprocessing 多进程的并行方法, 对脉冲星搜寻管线中的热点成功实现了多核并行, 并且设计了一个平衡负载的自动生成算法, 使用 multiprocessing+OpenMP 组合的方式, 解决了消色散算法中的负载不平衡问题, 进一步提升了总体的并行效率. 本文中针对多核架构的优化方法适用于 x86 和 ARM 平台, 并分别在两种平台上对 MWA-VCS 在中心频率为 185MHz 上的四个观测的非相干叠加数据进行了搜寻测试. 测试结论如下:

- 与原来的单线程方法相比, 在 28 个 CPU 核心的 x86 单机上的运行时间减少了 10.4–12.2 倍, 并行效率为 36%; 在 96 个 CPU 核心的 ARM 单机上的运行时间最多减少了 27.6 倍, 并行效率为 31%.
- 在 x86 和 ARM 计算节点上执行相同的并行优化技术, 展示了优化从 x86 到 ARM 平台的可移植性. 由于脉冲星搜索任务的大规模并行性,ARM 平台得益于每个节点更多的计算核数, 在测试程序中比 x86 平台的速度快 1.1–1.3 倍, 这个结果显示出 ARM 节点在 SKA 数据处理任务中的巨大潜力.
- 我们的优化算法使脉冲星搜索的时间消耗总体减少了一个数量级, 这将有效促进脉冲星研究的发展.

同时, 数据处理管线的性能优化不仅解决了脉冲星搜索管线的耗时问题, 还可以扩展到其他应用管线的性能优化, 对于中国 SKA 区域中心的天文研究有很大帮助（例如[48–55]）.

近几年内, 我们的工作将集中在进一步提高并行效率和向量化 PRESTO 管线上, 大量使用 MWA-VCS 的 SMART 巡天波束合成数据进行更大规模的实验, 以验证 ARM 平台的适用性. 对于即将到来的 SKA1 数据, 我们将做好充分的准备, 迎接脉冲星搜寻的挑战. 届时,ARM 平台将发挥其举足轻重的作用和影响.

致谢 本研究使用了中国 SKA 区域中心原型机的资源.

参考文献

1 Sturrock, P. A. "A model of pulsars." The Astrophysical Journal 164 (1971): 529.
2 Hewish, A., Bell, S. J., Pilkington, J. D. H., Scott, P. F., & Collins, R. A. Observation of a Rapidly Pulsating Radio Source. Nature, 1968, 217: 709
3 Cordes, J. M., Freire, P. C. C., Lorimer, D. R., et al. Arecibo Pulsar Survey Using ALFA. I. Survey Strategy and First Discoveries. The Astrophysical Journal, 2006, 637:446
4 Keith, M. J., Jameson, A., van Straten, W., et al. The High Time Resolution Universe Pulsar Survey - I. System configuration and initial discoveries. Monthly Notices of the Royal Astronomical Society, 2010, 409: 619

5 Barr, E. D., Champion, D. J., Kramer, M., et al. The Northern High Time Resolution Universe pulsar survey - I. Setup and initial discoveries. Monthly Notices of the Royal Astronomical Society, 2013, 435: 2234

6 Nan, R., Li, D., Jin, C., et al. The Five-Hundred Aperture Spherical Radio Telescope (FAST) Project. International Journal of Modern Physics D, 2011, 20: 6

7 Jiang, P., Tang, N.-Y., Hou, L.-G., et al. The fundamental performance of FAST with 19-beam receiver at L band. Research in Astronomy and Astrophysics, 2020, 20: 064

8 Qian, L., Yao, R., Sun, J., et al. 2020, The Innovation, 1, 100053

9 Cameron, A. D., Li, D., Hobbs, G., et al. An in-depth investigation of 11 pulsars discovered by FAST. Monthly Notices of the Royal Astronomical Society, 2020, 495: 3

10 Qian, L., Pan, Z., Li, D., et al. The first pulsar discovered by FAST. Science China Physics, Mechanics, and Astronomy, 2019, 62: 5

11 Pan, Z., Qian, L., Ma, X., et al. FAST Globular Cluster Pulsar Survey: Twenty-four Pulsars Discovered in 15 Globular Clusters. The Astrophysical Journal Letters, 2021, 915: 2

12 Han, J.-L., Wang, C., Wang, P.-F., et al. The FAST Galactic Plane Pulsar Snapshot survey: I. Project design and pulsar discoveries. Research in Astronomy and Astrophysics, 2021, 21: 107

13 Li, D., Wang, P., Qian, L., et al. FAST in Space: Considerations for a Multibeam, Multipurpose Survey Using China’s 500-m Aperture Spherical Radio Telescope (FAST). IEEE Microwave Magazine, 2018, 19: 3

14 Li, D., Dickey, J. M. & Liu, S. Preface: Planning the scientific applications of the Five-hundred-meter Aperture Spherical radio Telescope. Research in Astronomy and Astrophysics, 2019, 19: 2

15 Stovall, K., Lynch, R. S., Ransom, S. M., et al. The Green Bank Northern Celestial Cap Pulsar Survey. I. Survey Description, Data Analysis, and Initial Results. The Astrophysical Journal, 2014, 791: 67

16 J. S. Deneva et al., ”New discoveries from the Arecibo 327 MHz drift pulsar survey radio transient search”, The Astrophysical Journal, 2016, Volume 821, Number 1

17 http://www.naic.edu/ deneva/drift-search/

18 van Haarlem, M. P., Wise, M. W., Gunst, A. W., et al. LOFAR: The LOw-Frequency ARray. Astronomy and Astrophysics, 2013, 556: 2

19 Sanidas, S., Cooper, S., Bassa, C. G., et al. The LOFAR Tied-Array All-Sky Survey (LOTAAS): Survey overview and initial pulsar discoveries. Astronomy and Astrophysics, 2019, 626: A104

20 Tan, C. M., Bassa, C. G., Cooper, S., et al. The LOFAR Tied-Array all-sky survey: Timing of 21 pulsars including the first binary pulsar discovered with LOFAR. Monthly Notices of the Royal Astronomical Society, 2020, 492: 5878

21 Manchester, R. N., Hobbs, G. B., Teoh, A., & Hobbs, M. The Australia Telescope National Facility Pulsar Catalogue. The Astronomical Journal, 2005, 129: 1993

22 Keane, E., Bhattacharyya, B., Kramer, M., et al. A Cosmic Census of Radio Pulsars with the SKA. Advancing Astrophysics with the Square Kilometre Array (AASKA14), 2015,: 40

23 Qiuyu Yu.(2020) A PRESTO-based parallel pulsar search pipeline and pulsar signal de-noising, https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD202101&filename=1020968419.nh

24 Ransom, S. M. New search techniques for binary pulsars. Ph.D. Thesis, 2001

25 Sofia Dimoudi, Wesley Armour. Pulsar Acceleration Searches on the GPU for the Square Kilometre Array, https://arxiv.org/abs/1511.07343

26 Haomiao Wang and Prabu Thiagaraj and Oliver Sinnen. FPGA-based Acceleration of FT Convolution for Pulsar Search Using OpenCL[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2019, 11(4) : 1-25

27 Dewdney, P. E., Hall, P. J., Schilizzi, R. T., & Lazio, T. J. L. W. The Square Kilometre Array. IEEE Proceedings, 2009, 97: 1482

28 Tingay, S. J., Goeke, R., Bowman, J. D., et al. The Murchison Widefield Array: The Square Kilometre Array Precursor at Low Radio Frequencies. Publications of the Astronomical Society of Australia, 2013, 30: e007

29 Tremblay, S. E., Ord, S. M., Bhat, N. D. R., et al. The High Time and Frequency Resolution Capabilities of the Murchison Widefield Array. Publications of the Astronomical Society of Australia, 2015, 32, 5

30 Beardsley, A. P., Johnston-Hollitt, M., Trott, C. M., et al. Science with the Murchison Widefield Array: Phase I results and Phase II opportunities. Publications of the Astronomical Society of Australia, 2020, 37, 14

31 An, T. Science opportunities and challenges associated with SKA big data. Science China Physics, Mechanics, and Astronomy, 2019, 62: 989531

32 S. Ratcliffe, F. Graser, B. Carlson, O. B, SKA1 LOW SDP - CSP Interface Control Document, SKA Project Document number 100-000000-002 1 (1)

33 S. Ratcliffe, F. Graser, B. Carlson, O. B, SKA1 MID SDP - CSP Interface Control Document, SKA Project Document number 300-000000-002 1 (1)

34 S. Yamamura et al., "A64FX: 52-Core Processor Designed for the 442PetaFLOPS Supercomputer Fugaku," 2022 IEEE International Solid- State Circuits Conference (ISSCC), 2022, pp. 352-354, doi: 10.1109/ISSCC42614.2022.9731627

35 Peter Quinn, Michiel van Haarlem, Tao An, et al., "SKA Regional Centres - A White Paper by the SKA Regional Centre Steering Committee" V1.0, 2020

36 The SKA Regional Centre Steering Committee, SKA Regional Centres White Paper, v1.0, 2020 May.

37 T. An, X. P. Wu, and X. Y. Hong, SKA data take centre stage in China, Nature Astronomy, 3, 1030 (2019).

38 T.An, X.C.Wu, B.Q.Lao, et al. Status and progress of China SKA Regional Centre prototype. arxiv:2206.13022

39 B.Q.Lao, Y.K.Zhang, T.An, et al. Software Platform on China SKA Regional Center Prototype System(in Chinese).ChinaXiv:202206.00173.

40 J.W.Wei, C.F.Zhang, B.Q.Lao, et al. Optimization of parallel processing of Square Kilometre Array low frequency imaging pipeline (in Chinese). ChinaXiv:T202206.00292.

41 An, T., Wu, X. C., Lao, B. Q., et al. Status and progress of China SKA Regional Centre prototype. arxiv:2206.13022

42 ARM architecture, URL https://en.wikipedia.org/wiki/ARM_architecture, 2021.

43 Bhat, N. D. R., Swainston, N. A., McSweeney, S. J., et al. The Southern-sky MWA Rapid Two-metre (SMART) pulsar survey: System overview, pulsar census, and first pulsar discoveries, Publications of the Astronomical Society of Australia, 2022, in preparation

44 Ransom, S. M. PRESTO Home,
https://www.cv.nrao.edu/ sransom/presto/

45 Palach, Jan. Parallel programming with Python. Packt Publishing Ltd, 2014

46 Chandra, Rohit, et al. Parallel programming in OpenMP. Morgan kaufmann, 2001

47 Tabirca, Tatiana, et al. "A static workload balance scheduling algorithm." Proceedings. International Conference on Parallel Processing Workshop. IEEE, 2002

48 Gong, H., Zhang, Z., Xue, M., et al. Search and detection of northern pulsars in the side lobes of the murchison wide-field array. Scientia Sinica Physica, Mechanica & Astronomica, 2020, 50: 109501

49 Lao, B. Q., An, T., Yu, A., & Guo, S. G. "Research on Parallel Algorithms for uv-faceting Imaging" . Acta Astronomica Sinica, 2019, 60: 12

50 Lao, B., An, T., Yu, A., et al. "Parallel implementation of w-projection wide-field imaging" . Science Bulletin, 2019, 64, 586-594

51 Lao, B., An, T., Wang, A., et al. "Artificial intelligence for celestial object census: the latest technology meets the oldest science" . arXiv e-prints, 2021,: arXiv:2107.03082

52 An, T., Mohan, P., Zhang, Y., et al. Evolving parsec-scale radio structure in the most distant blazar known. Nature Communications, 2020, 11: 143

53 Gu, J., & Wang, J. Direct parameter inference from global EoR signal with Bayesian statistics. Monthly Notices of the Royal Astronomical Society, 2020, 492: 4080

54 Wang, R., Wicenec, A., & An, T. SKA shakes hands with Summit. Science Bulletin, 2020, 65: 337

55 Wang, Y., Tuntsov, A., Murphy, T., et al. ASKAP observations of multiple rapid scintillators reveal a degrees-long plasma filament. Monthly Notices of the Royal Astronomical Society, 2021, in press

Parallel optimization of the pulsar search pipeline on China SKA Regional Centre Prototype

Jianwen Wei[1], Chenfei Zhang[1], Zhongli Zhang[2*], Ting Yu[2], James Lin[1] & Tao An[2]

1. Shanghai Jiao Tong University,Shanghai,200240;
2. Shanghai Astronomical Observatory,Chinese Academy of Sciences,Shanghai,200030;
1. Shanghai Jiao Tong University,Shanghai 200240,China;
2. Shanghai Astronomical Observatory,Key Laboratory of Radio Astronomy,Chinese Academy of Sciences,Shanghai 200030,China

The connection between astronomy and high performance computing is becoming stronger with the development of cutting-edge observing facilities such as the Square Kilometre Array (SKA) and the proposed innovative platform for big data and high performance computing. Astronomical computation is characterized by huge data volume and massive parallelism, especially for pulsar search which is one of the leading scientific directions of the SKA. In this paper, we present an approach to accelerate the pulsar search pipeline based on OpenMP and multiprocessing techniques, propose a method to solve the load imbalance problem,and successfully has the pipeline installed on both x86 and ARM compute nodes on China SKA regional center prototype (CSRC-P). The performance evaluation from the tests on the Murchison Widefield Array (MWA) VCS observations shows that our optimization method works well on both x86 and ARM nodes, improving the relative speedup by a factor of 10.4–12.2 and 24.5–27.6, respectively, compared to the original single-thread approach. The ARM platform was found to be 1.1–1.3 times faster than the x86 platform in the tested cases, showing its great potential for SKA data processing. Recently, this optimized pulsar search pipeline deployed on CSRC-P will be especially used for low-frequency pulsar survey of the Southern-sky MWA Rapid Two-metre (SMART) program,for various scientific goals including pulsar timing arrays for gravitational wave detections.

Square Kilometre Array,pulsar,pulsar search,high performance computing,parallel optimization

PACS: 47.27.-i,47.27.Eq,47.27.Nz,47.40.Ki,47.85.Gj

doi: ??